EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	50	(US-6085296-$ or US-4819151-$ or US-5243698-$ or US-5414840-$ or US-5491359-$ or US-5506437-$ or US-5675763-$ or US-5784623-$ or US-5845331-$ or US-6003123-$ or US-4577274-$ or US-5182805-$ or US-5220661-$ or US-5247673-$ or US-5263161-$ or US-5274789-$ or US-5293486-$ or US-5301330-$ or US-5339449-$ or US-5394547-$ or US-5428810-$ or US-5430868-$ or US-5437031-$ or US-5488713-$ or US-5490276-$ or US-5513324-$).did. or (US-5560034-$ or US-5579482-$ or US-5630134-$ or US-5664088-$ or US-5668974-$ or US-5701470-$ or US-5706516-$ or US-5737529-$ or US-5748937-$ or US-5754779-$ or US-5761659-$ or US-5761670-$ or US-5809301-$ or US-5845325-$ or US-5864713-$ or US-5873127-$ or US-5892945-$ or US-5937187-$ or US-5935211-$ or US-5940610-$ or US-5951653-$ or US-5966547-$ or US-5978892-$ or US-6018785-$). did.	USPAT	OR	ON	2006/08/03 20:20
L2	15621	block near3 assign$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L3	8	L1 and L2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L4	5754620	process or processes or thread or threads	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L5	50	L1 and L4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L6	5396293	process or processes	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L7	50	L1 and L6	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L8	28165	memory adj block or memory adj blocks	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L9	7	L7 and L8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L10	2993971	block or blocks	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L11	7	L9 and L10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L12	15621	block near3 assign$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L13	5754620	process or processes or thread or threads	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L14	50	(US-6085296-$ or US-4819151-$ or US-5243698-$ or US-5414840-$ or US-5491359-$ or US-5506437-$ or US-5675763-$ or US-5784623-$ or US-5845331-$ or US-6003123-$ or US-4577274-$ or US-5182805-$ or US-5220661-$ or US-5247673-$ or US-5263161-$ or US-5274789-$ or US-5293486-$ or US-5301330-$ or US-5339449-$ or US-5394547-$ or US-5428810-$ or US-5430868-$ or US-5437031-$ or US-5488713-$ or US-5490276-$ or US-5513324-$).did. or (US-5560034-$ or US-5579482-$ or US-5630134-$ or US-5664088-$ or US-5668974-$ or US-5701470-$ or US-5706516-$ or US-5737529-$ or US-5748937-$ or US-5754779-$ or US-5761659-$ or US-5761670-$ or US-5809301-$ or US-5845325-$ or US-5864713-$ or US-5873127-$ or US-5892945-$ or US-5937187-$ or US-5935211-$ or US-5940610-$ or US-5951653-$ or US-5966547-$ or US-5978892-$ or US-6018785-$). did.	USPAT	OR	ON	2006/08/03 20:20
L15	50	L14 and L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L16	5396293	process or processes	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L17	50	L14 and L16	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L18	28165	memory adj block or memory adj blocks	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L19	2993971	block or blocks	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L20	50	(US-6085296-$ or US-4819151-$ or US-5243698-$ or US-5414840-$ or US-5491359-$ or US-5506437-$ or US-5675763-$ or US-5784623-$ or US-5845331-$ or US-6003123-$ or US-4577274-$ or US-5182805-$ or US-5220661-$ or US-5247673-$ or US-5263161-$ or US-5274789-$ or US-5293486-$ or US-5301330-$ or US-5339449-$ or US-5394547-$ or US-5428810-$ or US-5430868-$ or US-5437031-$ or US-5488713-$ or US-5490276-$ or US-5513324-$).did. or (US-5560034-$ or US-5579482-$ or US-5630134-$ or US-5664088-$ or US-5668974-$ or US-5701470-$ or US-5706516-$ or US-5737529-$ or US-5748937-$ or US-5754779-$ or US-5761659-$ or US-5761670-$ or US-5809301-$ or US-5845325-$ or US-5864713-$ or US-5873127-$ or US-5892945-$ or US-5937187-$ or US-5935211-$ or US-5940610-$ or US-5951653-$ or US-5966547-$ or US-5978892-$ or US-6018785-$). did.	USPAT	OR	ON	2006/08/03 20:20
L21	8	L14 and L12	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L22	7	L17 and L18	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L23	7	L22 and L19	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L24	7266	L16 with communication with (manage or managed or managing or management)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L25	2897635	process.ti. or processes.ti. or process.ab. or processes.ab.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L26	3227	L24 and L25	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L27	846	set adj3 ((memory adj block) or (memory adj blocks))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L28	3	L26 and L27	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L29	14	(set adj2 ((memory adj block) or (memory adj blocks))) with (assign or assigned or assigning)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L30	24307724	@ad<"20030630"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L31	9	L29 and L30	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L32	782	interprocess adj communication?	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L33	21274	shared adj memory	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L34	128	L32 same L33	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L35	11	L27 and L34	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L36	4	L30 and L35	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L37	279	(buddy adj system) or (heap adj management)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L38	2	L34 and L37	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L39	44741	data near3 request?	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L40	2928	(data near3 request?) near5 (process or processes or thread or threads)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L41	0	L37 and L40	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L42	18	L32 and L40	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L43	0	L15 and L42	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L44	0	L27 and L42	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L45	0	L18 and L42	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L46	0	L37 and L42	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L47	12	L33 and L42	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L48	11	L30 and L47	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L49	240	assigned adj memory adj block	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L50	3	L13 near3 L49	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L51	3	(US-6442596-$ or US-5692157-$ or US-5684945-$).did.	USPAT	OR	ON	2006/08/03 20:20
L52	0	cache with (assign or assigned or assigning) with L18 with L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L53	144	cache with L18 with L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L54	2	memory adj manager same L53	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L55	1914	data adj access adj request	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L56	10748	allocate adj memory or memory adj allocation	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L57	0	L55 with L56 with L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L58	10	(request or requests) with assign$4 with memory adj block with L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L59	140	L40 and L56 and L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L60	3143	interprocess adj communication	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L61	2912	L56 same L13	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L62	144	L60 and L61	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L63	2	L49 and L62	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L64	4	L34 and L40	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L65	190	dynamic near3 block near3 allocation	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L66	157	L13 and L65	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L67	34	L55 and L66	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L68	33	L27 and L60	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L69	1103	711/147.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L70	1	L68 and L69	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L71	1527	711/202.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L72	2270	711/154.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L73	168	719/312.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L74	4846	L69 or L71 or L72 or L73	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L75	79	L40 and L74	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L76	0	L32 and L75	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L77	0	L37 and L75	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L78	0	L37 and "6344" and L75	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L79	8	L56 and L75	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L80	6	L30 and L79	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L81	4	(US-20020013889-$).did. or (US-6591355-$ or US-6112281-$ or US-5566315-$).did.	US-PGPUB; USPAT	OR	ON	2006/08/03 20:20
L82	1	L40 and L73 and L30	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L83	0	L58 and L73 and L30	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L84	9	L14 and L56	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L85	0	L14 and L58	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L86	0	L14 and L55	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L87	0	L14 and L65	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L88	9922	L60 or ipc	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20
L89	7	L14 and L88	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:20

EAST Search History

L90	689	cache adj module	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:56
L91	22	buddy adj system near5 allocat$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:59
L92	1	32 and 90	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 20:59
L93	2	32 and 91	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:00
L94	12	60 and 65	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:01
L95	11	30 and 94	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:03
L96	639	assign$3 near5 memory adj block	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:03
L97	95	(assign$3 near5 memory adj block) same (process or processes)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:04
L98	1914	data adj access adj request	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:04
L99	4	97 and 98	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:04
L100	0	30 and 99	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2006/08/03 21:07

EAST Search History

L101	199	711/147.ccls.	US-PGPUB	OR	ON	2006/08/03 21:07
L102	1	97 and 101	US-PGPUB	OR	ON	2006/08/03 21:08
L103	133	30 and 101	US-PGPUB	OR	ON	2006/08/03 21:09
L104	3	40 and 103	US-PGPUB	OR	ON	2006/08/03 21:09

Terms used **dynamic memory allocation** and **memory blocks** and **request** Found **50,726** of **175,083**

Sort results by relevance Save results to a Binder Try an Advanced Search

Display results expanded form Search Tips Try this search in The ACM Guide

Open results in a new window

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Dynamic memory allocation in computer simulation

Norman R. Nielsen

November 1977 **Communications of the ACM**, Volume 20 Issue 11

Publisher: ACM Press

Full text available: pdf(1.01 MB) Additional Information: full citation, abstract, references, citings

e of 35 dynamic memory allocation algorithms when used to service simulation programs as represented by 18 test cases. Algorithm performance was measured in terms of processing time, memory usage, and external memory fragmentation. Algorithms maintaining separate free space lists for each size of memory block used tended to perform quite well compared with other algorithms. Simple algorithms operating on memory ordered lists (without any free list) performed surprisingly well. Algorithms employi ...

Keywords: algorithm performance, dynamic memory allocation, dynamic memory management, dynamic storage allocation, garbage collection, list processing, memory allocation, memory management, programming techniques, simulation, simulation memory management, simulation techniques, space allocation, storage allocation

2 A dynamic memory management unit for embedded real-time system-on-a-chip

Mohamed Shalan, Vincent J. Mooney

November 2000 **Proceedings of the 2000 international conference on Compilers, architecture, and synthesis for embedded systems**

Publisher: ACM Press

Full text available: pdf(321.80 KB) Additional Information: full citation, citings

Keywords: SoCDMMU, dynamic memory management, embedded systems, real-time systems, system-on-a-chip, two-level memory management

3 General topics in computer science I - GCS I: Variations on the binary buddy system for dynamic memory management

Arie Kaufman

March 1980 **Proceedings of the 18th annual Southeast regional conference**

Publisher: ACM Press

Full text available: pdf(393.81 KB) Additional Information: full citation, abstract, references, citings

Two new variations of the binary buddy system for dynamic storage bookkeeping are introduced. These variations, the revised buddy system and the tailored-lists buddy system, recombine smaller memory blocks only upon necessity in an attempt to save on processor execution time. Simulation results comparing the three systems reveal that the two new variations are superior to the original buddy system, since they are faster and the difference in memory utilization is insignificant. A comparison of t ...

Keywords: binary buddy system, dynamic storage allocation, fragmentation, revised buddy system, tailored-lists buddy system

4 Dynamic verification of operating system decisions

R. S. Fabry

November 1973

Communications of the ACM, Volume 16 Issue 11

Publisher: ACM Press

Full text available: pdf(1.09 MB) Additional Information: full citation, abstract, references, citings

Dynamic verification of a decision implies that every time the decision is made there is a consistency check performed on the decision using independent hardware and software. The dynamic verification of operating system decisions is used on the PRIME system being designed and constructed at the University of California, Berkeley. PRIME is an experimental time-sharing system which is to have the properties of continuous availability, data privacy, and cost effectiveness. The technique of dy ...

Keywords: data privacy, data security, fault tolerance, modular computer systems, operating systems, program verification, software reliability

5 Design space exploration and architectural design of HW/SW systems: Hardware support for real-time embedded multiprocessor system-on-a-chip memory management

Mohamed Shalan, Vincent J. Mooney

May 2002 **Proceedings of the tenth international symposium on Hardware/software codesign**

Publisher: ACM Press

Full text available: pdf(533.74 KB) Additional Information: full citation, abstract, references, index terms, review

The aggressive evolution of the semiconductor industry --- smaller process geometries, higher densities, and greater chip complexity --- has provided design engineers the means to create complex high-performance Systems-on-a-Chip (SoC) designs. Such SoC designs typically have more than one processor and huge memory, all on the same chip. Dealing with the global on- chip memory allocation/de-allocation in a dynamic yet deterministic way is an important issue for the upcoming billion transistor mu ...

Keywords: Atalanta, SoCDMMU, System-on-a-Chip, dynamic memory management, embedded systems, real-time operating systems., real-time systems, two-level memory management

6 Dynamic Memory Management Design Methodology for Reduced Memory Footprint in Multimedia and Wireless Network Applications

David Atienza, Stylianos Mamagkakis, Francky Catthoor, Jose M. Mendias, Dimitris Soudris

February 2004 **Proceedings of the conference on Design, automation and test in Europe - Volume 1**

Publisher: IEEE Computer Society

Full text available: pdf(154.12 KB) Additional Information: full citation, abstract, index terms

New portable consumer embedded devices must execute multimedia and wireless network applications that demand extensive memory footprint. Moreover, they must heavily rely on Dynamic Memory (DM) due to the unpredictability of the input data (e.g. 3D streams features) and system behaviour (e.g. number of applications running concurrently defined by the user). Within this context, consistent design methodologies that can tackle ef.ciently the complex DM behaviour of these multimedia and network appl ...

7 A weighted buddy method for dynamic storage allocation

Kenneth K. Shen, James L. Peterson

October 1974 **Communications of the ACM**, Volume 17 Issue 10

Publisher: ACM Press

Full text available: pdf(461.71 KB) Additional Information: full citation, abstract, references, citings, index terms

An extension of the buddy method, called the weighted buddy method, for dynamic storage allocation is presented. The weighted buddy method allows block sizes of 2k and 3·2k, whereas the original buddy method allowed only block sizes of 2k. This extension is achieved at an additional cost of only two bits per block. Simulation results are presented which compare ...

Keywords: buddy system, dynamic storage allocation, memory allocation, weighted buddy algorithm

8 Efficient implementation of the first-fit strategy for dynamic storage allocation

R. P. Brent

July 1989 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 11 Issue 3

Publisher: ACM Press

Full text available: pdf(1.05 MB) Additional Information: full citation, abstract, references, citings, index terms, review

We describe an algorithm that efficiently implements the first-fit strategy for dynamic storage allocation. The algorithm imposes a storage overhead of only one word per allocated block (plus a few percent of the total space used for dynamic storage), and the time required to allocate or free a block is O(log W), where W is the maximum number of words allocated dynamically. The algorithm is faster than many commonly used algorithms, especia ...

9 CAD 2: Power-aware RAM mapping for FPGA embedded memory blocks

Russell Tessier, Vaughn Betz, David Neto, Thiagaraja Gopalsamy

February 2006 **Proceedings of the internation symposium on Field programmable gate arrays FPGA'06**

Publisher: ACM Press

Full text available: pdf(151.01 KB) Additional Information: full citation, abstract, references, index terms

Embedded memory blocks are important resources in contemporary FPGA devices. When targeting FPGAs, application designers often specify high-level memory functions which exhibit a range of sizes and control structures. These logical memories must be mapped to FPGA embedded memory resources such that physical design objectives are met. In this work a set of power-aware logical-to-physical RAM mapping algorithms are described which convert user-defined memory specifications to on-chip FPGA memory b ...

Keywords: FPGA, dynamic power, embedded memory block

10 Evaluating the memory overhead required for COMA architectures

T. Joe, J. L. Hennessy

April 1994 **ACM SIGARCH Computer Architecture News , Proceedings of the 21ST annual international symposium on Computer architecture ISCA '94**, Volume 22 Issue 2

Publisher: IEEE Computer Society Press, ACM Press

Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, citings, index terms

Cache only memory architectures (COMA) have an inherent memory overhead due to the organization of main memory as a large cache called an attraction memory. This overhead consists of memory left unallocated for performance reasons as well as additional physical memory required due to the cache organization of memory. In this work, we examine the effect of data reshuffling and data replication on the memory overhead. Data reshuffling occurs when space needs to be allocated to store a remote memor ...

11 Cache coherence in large-scale shared-memory multiprocessors: issues and comparisons

David J. Lilja

September 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 3

Publisher: ACM Press

Full text available: pdf(3.12 MB) Additional Information: full citation, references, citings, index terms

12 Data remapping for design space optimization of embedded memory systems

Rodric M. Rabbah, Krishna V. Palem

May 2003 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 2 Issue 2

Publisher: ACM Press

Full text available: pdf(885.05 KB) Additional Information: full citation, abstract, references, citings, index terms

In this article, we present a novel linear time algorithm for *data remapping*, that is, (i) lightweight; (ii) fully automated; and (iii) applicable in the context of pointer-centric programming languages with dynamic memory allocation support. All previous work in this area lacks one or more of these features. We proceed to demonstrate a *novel application of this algorithm as a key step in optimizing the design of an embedded memory system.* Specifically, we show that by virtue of lo ...

Keywords: Design space exploration, caches, compiler optimization, data remapping, embedded systems, memory hierarchy, memory subsystem

13 Creating and preserving locality of java applications at allocation and garbage collection times

Yefim Shuf, Manish Gupta, Hubertus Franke, Andrew Appel, Jaswinder Pal Singh

November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '02**, Volume 37 Issue 11

Publisher: ACM Press

Full text available: pdf(180.20 KB) Additional Information: full citation, abstract, references, citings, index terms

The growing gap between processor and memory speeds is motivating the need for optimization strategies that improve data locality. A major challenge is to devise techniques suitable for pointer-intensive applications. This paper presents two techniques aimed at improving the memory behavior of pointer-intensive applications with dynamic memory allocation, such as those written in Java. First, we present an allocation time object placement technique based on the recently introduced notion of *p* ...

Keywords: *JVM, Java, garbage collection, heap traversal, locality, locality based graph traversal, memory allocation, memory management, object co-allocation, object placement, prolific types, run-time systems*

14 A unified model of pointwise equivalence of procedural computations

David G. von Bank, Charles M. Shub, Robert W. Sebesta

November 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 6

Publisher: ACM Press

Full text available: pdf(2.10 MB) Additional Information: full citation, abstract, references, citings, index terms, review

The execution of a program on a processor is viewed as a representation of that program going through a sequence of states. Each state change is manifested by the execution of a single instruction. Models that depend on this perspective are presented. The first is a static model of a description of a procedural computation. This model formalizes the description of the information in an executable module. Following this dynamic model of a procedural computation is given. This second model de ...

Keywords: dynamic migration, heterogeneous migration

15 Memory and network optimization in embedded designs: An integrated hardware/software approach for run-time scratchpad management

Poletti Francesco, Paul Marchal, David Atienza, Luca Benini, Francky Catthoor, Jose M. Mendias

June 2004 **Proceedings of the 41st annual conference on Design automation**

Publisher: ACM Press

Full text available: pdf(137.98 KB) Additional Information: full citation, abstract, references, citings, index terms

An ever increasing number of dynamic interactive applications are implemented on portable consumer electronics. Designers depend largely on operating systems to map these applications on the architecture. However, today's embedded operating systems abstract away the precise architectural details of the platform. As a consequence, they cannot exploit the energy efficiency of scratchpad memories. We present in this paper a novel integrated hardware/software solution to support scratchpad memories ...

Keywords: AMBA AHB, DMA, dynamic allocation, scratchpad

16 The memory fragmentation problem: solved?

Mark S. Johnstone, Paul R. Wilson

October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3

Publisher: ACM Press

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

We show that for 8 real and varied C and C++ programs, several conventional dynamic storage allocators provide near-zero fragmentation, once we account for overheads due to implementation details such as headers, alignment, etc. This substantially strengthens our previous results showing that the memory fragmentation problem has generally been misunderstood, and that good allocator policies can provide good memory usage for most programs. The new results indicate that for most programs, excellen ...

17 Memory sharing predictor: the key to a speculative coherent DSM

An-Chow Lai, Babak Falsafi

May 1999 **ACM SIGARCH Computer Architecture News , Proceedings of the 26th annual international symposium on Computer architecture ISCA '99**, Volume 27 Issue 2

Publisher: IEEE Computer Society, ACM Press

Full text available: pdf(122.70 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Recent research advocates using general message predictors to learn and predict the coherence activity in distributed shared memory (DSM). By accurately predicting a message and timely invoking the necessary coherence actions, a DSM can hide much of the remote access latency. This paper proposes the *Memory Sharing Predictors (MSPs),* pattern-based predictors that significantly improve prediction accuracy and implementation cost over general message predictors. An MSP is based on the key ob ...

18 Advanced memory allocation

Gianluca Insolvibile

May 2003 **Linux Journal**, Volume 2003 Issue 109

Publisher: Specialized Systems Consultants, Inc.

Full text available: html(15.61 KB) Additional Information: full citation, abstract, index terms

Keeping memory requirements low cansave you time and money. Here's how to bend malloc() to your will.

19 Software-extended coherent shared memory: performance and cost

D. Chaiken, A. Agarwal
April 1994 **ACM SIGARCH Computer Architecture News , Proceedings of the 21ST annual international symposium on Computer architecture ISCA '94**, Volume 22 Issue 2
Publisher: IEEE Computer Society Press, ACM Press
Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper evaluates the tradeoffs involved in the design of the software-extended memory system of Alewife, a multiprocessor architecture that implements coherent shared memory through a combination of hardware and software mechanisms. For each block of memory, Alewife implements between zero and five coherence directory pointers in hardware and allows software to handle requests when the pointers are exhausted. The software includes a flexible coherence interface that facilitates protocol soft ...

20 Buddy systems
James L. Peterson, Theodore A. Norman
June 1977 **Communications of the ACM**, Volume 20 Issue 6
Publisher: ACM Press
Full text available: pdf(971.53 KB) Additional Information: full citation, abstract, references, citings

Two algorithms are presented for implementing any of a class of buddy systems for dynamic storage allocation. Each buddy system corresponds to a set of recurrence relations which relate the block sizes provided to each other. Analyses of the internal fragmentation of the binary buddy system, the Fibonacci buddy system, and the weighted buddy system are given. Comparative simulation results are also presented for internal, external, and total fragmentation.

Keywords: Fibonacci buddy system, buddy system, dynamic storage allocation, fragmentation, weighted buddy system

Terms used **dynamic memory allocation** and **memory blocks** and **request** Found **50,726** of **175,083**

Sort results by relevance Save results to a Binder Try an Advanced Search

Display results expanded form Search Tips Try this search in The ACM Guide

Open results in a new window

Results 21 - 40 of 200 Result page: previous 1 **2** 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

21 Algorithmic generality in memory management design

Dinesh Kumar

April 1974 **ACM SIGSIM Simulation Digest**, Volume 5 Issue 3

Publisher: ACM Press

Full text available: pdf(1.23 MB) Additional Information: full citation, abstract, references

As computer systems grew in complexity, the area of memory management has become increasingly important. In future operating systems what is needed is a generalized, flexible, modular, and open ended-design of the functional modules (e.g. memory management modules). To achieve these properties, sufficient ingenuity and effort are required during both design and implementation. This paper outlines an approach which should lead to a design having these properties. A memory management system was de ...

22 Architectural primitives for a scalable shared memory multiprocessor

Joonwon Lee, Umakishore Ramachandran

June 1991 **Proceedings of the third annual ACM symposium on Parallel algorithms and architectures**

Publisher: ACM Press

Full text available: pdf(1.27 MB) Additional Information: full citation, references, citings, index terms

23 Efficient management for large-scale flash-memory storage systems with resource conservation

Li-Pin Chang, Tei-Wei Kuo

November 2005 **ACM Transactions on Storage (TOS)**, Volume 1 Issue 4

Publisher: ACM Press

Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, index terms

Many existing approaches on flash-memory management are based on RAM-resident tables in which one single granularity size is used for both address translation and space management. As high-capacity flash memory is becoming more affordable than ever, the dilemma of how to manage the RAM space or how to improve the access performance is emerging for many vendors. In this article, we propose a tree-based management scheme which adopts multiple granularities in flash-memory management. Our objective ...

Keywords: Flash memory, consumer electronics, embedded systems, memory management, portable devices, storage systems

24 Virtual memory and backing storage management in multiprocessor operating systems using object-oriented design techniques

V. F. Russo, R. H. Campbell

September 1989 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications OOPSLA '89**, Volume 24 Issue 10

Publisher: ACM Press

Full text available: pdf(1.19 MB) Additional Information: full citation, abstract, references, citings, index terms

The Choices operating system architecture [3, 4, 15] uses class hierarchies and object-oriented programming to facilitate the construction of customized operating systems for shared memory and networked multiprocessors. The software is being used in the Tapestry Parallel Computing Laboratory at the University of Illinois to study the performance of algorithms, mechanisms, and policies for parallel systems. This paper describes the architectural design and class hierarchy of ...

25 Design space optimization of embedded memory systems via data remapping

Krishna V. Palem, Rodric M. Rabbah, Vincent J. Mooney, Pinar Korkmaz, Kiran Puttaswamy

June 2002 **ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems LCTES/SCOPES '02**, Volume 37 Issue 7

Publisher: ACM Press

Full text available: pdf(382.47 KB) Additional Information: full citation, abstract, references, citings, index terms

In this paper, we provide a novel compile-time *data remapping* algorithm that runs in linear time. This remapping algorithm is the first fully automatic approach applicable to pointer-intensive dynamic applications. We show that data remapping can be used to significantly reduce the *energy consumed* as well as the *memory size* needed to meet a user-specified performance goal (i.e., execution time) -- relative to the same application executing without being remapped. These twin ...

Keywords: data remapping, design space exploration, power aware

26 Efficient strategies for software-only protocols in shared-memory multiprocessors

Håkan Grahn, Per Stenström

May 1995 **ACM SIGARCH Computer Architecture News , Proceedings of the 22nd annual international symposium on Computer architecture ISCA '95**, Volume 23 Issue 2

Publisher: ACM Press

Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, citings, index terms

The cost, complexity, and inflexibility of hardware-based directory protocols motivate us to study the performance implications of protocols that emulate directory management using software handlers executed on the compute processors. An important performance limitation of such software-only protocols is that software latency associated with directory management ends up on the critical memory access path for read miss transactions. We propose five strategies that support efficient data transfers ...

27 Tempest and typhoon: user-level shared memory

S. K. Reinhardt, J. R. Larus, D. A. Wood

April 1994 **ACM SIGARCH Computer Architecture News , Proceedings of the 21ST annual international symposium on Computer architecture ISCA '94**, Volume 22 Issue 2

Publisher: IEEE Computer Society Press, ACM Press

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

Future parallel computers must efficiently execute not only hand-coded applications but also programs written in high-level, parallel programming languages. Today's machines limit these programs to a single communication paradigm, either message-passing or shared-memory, which results in uneven performance. This paper addresses this problem by defining an interface, *Tempest*, that exposes low-level communication and memory-system mechanisms so programmers and compilers can customize polici ...

28 Tempest and typhoon: user-level shared memory

Steven K. Reinhardt, James R. Larus, David A. Wood

August 1998 **25 years of the international symposia on Computer architecture (selected papers)**

Publisher: ACM Press

Full text available: pdf(1.57 MB) Additional Information: full citation, references, index terms

29 Using prediction to accelerate coherence protocols

Shubhendu S. Mukherjee, Mark D. Hill

April 1998 **ACM SIGARCH Computer Architecture News , Proceedings of the 25th annual international symposium on Computer architecture ISCA '98**, Volume 26 Issue 3

Publisher: IEEE Computer Society, ACM Press

Full text available: pdf(1.71 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Most large shared-memory multiprocessors use directory protocols to keep per-processor caches coherent. Some memory references in such systems, however, suffer long latencies for misses to remotely-cached blocks. To ameliorate this latency, researchers have augmented standard coherence protocols with optimizations for specific sharing patterns, such as read-modify-write, producer-consumer, and migratory sharing. This paper seeks to replace these directed solutions with general prediction logic t ...

30 Tuning garbage collection for reducing memory system energy in an embedded java environment

G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, M. Wolczko

November 2002 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 1 Issue 1

Publisher: ACM Press

Full text available: pdf(740.23 KB) Additional Information: full citation, abstract, references, citings, index terms

Java has been widely adopted as one of the software platforms for the seamless integration of diverse computing devices. Over the last year, there has been great momentum in adopting Java technology in devices such as cellphones, PDAs, and pagers where optimizing energy consumption is critical. Since, traditionally, the Java virtual machine (JVM), the cornerstone of Java technology, is tuned for performance, taking into account energy consumption requires reevaluation, and possibly redesign of t ...

Keywords: Garbage collector, Java Virtual Machine (JVM), K Virtual Machine (KVM), low power computing

31 Operating systems for sensor networks: A dynamic operating system for sensor nodes

Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, Mani Srivastava

June 2005 **Proceedings of the 3rd international conference on Mobile systems, applications, and services MobiSys '05**

Publisher: ACM Press

Full text available: pdf(418.07 KB) Additional Information: full citation, abstract, references

Sensor network nodes exhibit characteristics of both embedded systems and general-purpose systems. They must use little energy and be robust to environmental conditions, while also providing common services that make it easy to write applications. In TinyOS, the current state of the art in sensor node operating systems, reusable components implement common services, but each node runs a single statically-linked system image, making it hard to run multiple applications or incrementally update app ...

32 Options for dynamic address translation in COMAs

Xiaogang Qiu, Michel Dubois

April 1998 **ACM SIGARCH Computer Architecture News , Proceedings of the 25th annual international symposium on Computer architecture ISCA '98**, Volume 26 Issue 3

Publisher: IEEE Computer Society, ACM Press

Full text available: pdf(1.37 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

In modern processors, the dynamic translation of virtual addresses to support virtual memory is done before or in parallel with the first-level cache access. As processor technology improves at a rapid pace and the working sets of new applications grow insatiably the latency and bandwidth demands on the TLB (Translation Lookaside Buffer) are getting more and more difficult to meet. The situation is worse in multiprocessor systems, which run larger applications and are plagued by the TLB consiste ...

33 Computational processor demands of Algol-60 programs

Robert E. Brundage, Alan P. Batson

November 1975 **Proceedings of the fifth ACM symposium on Operating systems principles**

Publisher: ACM Press

Full text available: pdf(712.93 KB) Additional Information: full citation, abstract, references, citings, index terms

The characteristics of computational processor requirements of a sample of Algol-60 programs have been measured. Distributions are presented for intervals of processor activity as defined by input-output requests and segment allocation requests occurring within the Johnston contour model and within a stack model in which array allocations are treated separately. The results provide new empirical data concerning the behavior of this class of programs. Some implications of the empirical resul ...

Keywords: Algol-60, Contour model, Processor distributions, Program behavior, Resource allocation

34 Composing high-performance memory allocators

Emery D. Berger, Benjamin G. Zorn, Kathryn S. McKinley

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation PLDI '01**, Volume 36 Issue 5

Publisher: ACM Press

Full text available: pdf(1.56 MB) Additional Information: full citation, references, citings, index terms

35 Symbolic bounds analysis of pointers, array indices, and accessed memory regions

Radu Rugina, Martin C. Rinard

March 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 2

Publisher: ACM Press

Full text available: pdf(490.56 KB) Additional Information: full citation, abstract, references, index terms

This article presents a novel framework for the symbolic bounds analysis of pointers, array indices, and

accessed memory regions. Our framework formulates each analysis problem as a system of inequality constraints between symbolic bound polynomials. It then reduces the constraint system to a linear program. The solution to the linear program provides symbolic lower and upper bounds for the values of pointer and array index variables and for the regions of memory that each statement and procedur ...

Keywords: Symbolic analysis, parallelization, static race detection

36 A taxonomy-based comparison of several distributed shared memory systems

Ming-Chit Tam, Jonathan M. Smith, David J. Farber

July 1990 **ACM SIGOPS Operating Systems Review**, Volume 24 Issue 3

Publisher: ACM Press

Full text available: pdf(1.96 MB) Additional Information: full citation, abstract, citings, index terms

Two possible modes of Input/Output (I/O)are "sequential" and "random-access", and there is an extremely strong conceptual link between I/O and communication. Sequential communication, typified in the I/O setting by magnetic tape, is typified in the communication setting by a **stream**, e.g., a UNIX[1] pipe. Random-access communication, typified in the I/O setting by a drum or disk device, is typified in the communication setting by **shared memory**. In this paper, we study and s ...

37 A serializability violation detector for shared-memory server programs

Min Xu, Rastislav Bodík, Mark D. Hill

June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation PLDI '05**, Volume 40 Issue 6

Publisher: ACM Press

Full text available: pdf(326.11 KB) Additional Information: full citation, abstract, references, index terms

We aim to improve reliability of multithreaded programs by proposing a dynamic detector that detects potentially erroneous program executions and their causes. We design and evaluate a *Serializability Violation Detector* (SVD) that has two unique goals: (I) triggering automatic recovery from erroneous executions using backward error recovery (BER), or simply alerting users that a software error may have occurred; and (II) helping debug programs by revealing causes of error symptoms.Two pro ...

Keywords: multithreading, race conditions, serializability

38 Avoiding initialization misses to the heap

Jarrod A. Lewis, Bryan Black, Mikko H. Lipasti

May 2002 **ACM SIGARCH Computer Architecture News , Proceedings of the 29th annual international symposium on Computer architecture ISCA '02 , Proceedings of the 29th annual international symposium on Computer architecture ISCA '02**, Volume 30 Issue 2

Publisher: IEEE Computer Society, ACM Press

Full text available: pdf(1.29 MB) Publisher Site Additional Information: full citation, abstract, references, index terms

This paper investigates a class of main memory accesses (*invalid memory traffic*) that can be eliminated altogether. Invalid memory traffic is real data traffic that transfers invalid data. By tracking the initialization of dynamic memory allocations, it is possible to identify store instructions that miss the cache and would fetch uninitialized heap data. The data transfers associated with these initialization misses can be avoided without losing correctness. The memory system property cr ...

Keywords: invalid memory traffic, initializing stores, cache installation, allocation range cache

39 Shasta: a low overhead, software-only approach for supporting fine-grain shared memory

Daniel J. Scales, Kourosh Gharachorloo, Chandramohan A. Thekkath

October 1996 **ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of the seventh international conference on Architectural support for programming languages and operating systems ASPLOS-VII**, Volume 30 , 31 Issue 5 , 9

Publisher: ACM Press

Full text available: pdf(1.49 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes Shasta, a system that supports a shared address space in software on clusters of computers with physically distributed memory. A unique aspect of Shasta compared to most other software distributed shared memory systems is that shared data can be kept coherent at a fine granularity. In addition, the system allows the coherence granularity to vary across different shared data structures in a single application. Shasta implements the shared address space by transparently rewrit ...

40 An efficient cache-based access anomaly detection scheme

Sang L. Min, Jong-Deok Choi

April 1991 **ACM SIGARCH Computer Architecture News , ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of the fourth international conference on Architectural support for programming languages and operating systems ASPLOS-IV,** Volume 19 , 25 , 26 Issue 2 , Special Issue , 4

Publisher: ACM Press

Full text available: pdf(1.20 MB) Additional Information: full citation, references, citings, index terms

Terms used **thread** and **memory management** and **allocation** and **deallocation**

Found **7,192** of **167,655**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Nonblocking memory management support for dynamic-sized data structures

Maurice Herlihy, Victor Luchangco, Paul Martin, Mark Moir

May 2005 **ACM Transactions on Computer Systems (TOCS)**, Volume 23 Issue 2

Publisher: ACM Press

Full text available: pdf(944.89 KB) Additional Information: full citation, abstract, references, index terms

Conventional dynamic memory management methods interact poorly with lock-free synchronization. In this article, we introduce novel techniques that allow lock-free data structures to allocate and free memory dynamically using any thread-safe memory management library. Our mechanisms are lock-free in the sense that they do not allow a thread to be prevented from allocating or freeing memory by the failure or delay of other threads. We demonstrate the utility of these techniques by showing how to m ...

Keywords: Multiprocessors, concurrent data structures, dynamic data structures, memory management, nonblocking synchronization

2 Compiling nested data-parallel programs for shared-memory multiprocessors

Siddhartha Chatterjee

July 1993 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 15 Issue 3

Publisher: ACM Press

Full text available: pdf(4.17 MB) Additional Information: full citation, references, citings, index terms, review

Keywords: compilers, data parallelism, shared-memory multiprocessors

3 Designing a trace format for heap allocation events

Trishul Chilimbi, Richard Jones, Benjamin Zorn

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1

Publisher: ACM Press

Full text available: pdf(1.53 MB) Additional Information: full citation, abstract, citings, index terms

Dynamic storage allocation continues to play an important role in the performance and correctness of systems ranging from user productivity software to high-performance servers. While algorithms for dynamic storage allocation have been studied for decades, much of the literature is based on measuring the performance of benchmark programs

unrepresentative of many important allocation-intensive workloads. Furthermore, to date no standard has emerged or been proposed for publishing and exchangin ...

4 Revising old friends: Capriccio: scalable threads for internet services

Rob von Behren, Jeremy Condit, Feng Zhou, George C. Necula, Eric Brewer

October 2003 **Proceedings of the nineteenth ACM symposium on Operating systems principles**

Publisher: ACM Press

Full text available: pdf(312.83 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents Capriccio, a scalable thread package for use with high-concurrency servers. While recent work has advocated event-based systems, we believe that thread-based systems can provide a simpler programming model that achieves equivalent or superior performance.By implementing Capriccio as a user-level thread package, we have decoupled the thread package implementation from the underlying operating system. As a result, we can take advantage of cooperative threading, new asynchronous ...

Keywords: blocking graph, dynamic stack growth, linked stack management, resource-aware scheduling, user-level threads

5 Escape analysis for Java™: Theory and practice

Bruno Blanchet

November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6

Publisher: ACM Press

Full text available: pdf(684.21 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Escape analysis is a static analysis that determines whether the lifetime of data may exceed its static scope.This paper first presents the design and correctness proof of an escape analysis for Java™. This analysis is interprocedural, context sensitive, and as flow-sensitive as the static single assignment form. So, assignments to object fields are analyzed in a flow-insensitive manner. Since Java is an imperative language, the effect of assignments must be precisely determined. Thi ...

Keywords: Java, optimization, stack allocation, static analysis, synchronization elimination

6 The space cost of lazy reference counting

Hans-J. Boehm

January 2004 **ACM SIGPLAN Notices , Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL '04**, Volume 39 Issue 1

Publisher: ACM Press

Full text available: pdf(125.88 KB) Additional Information: full citation, abstract, references, index terms

Reference counting memory management is often advocated as a technique for reducing or avoiding the pauses associated with tracing garbage collection. We present some measurements to remind the reader that classic reference count implementations may in fact exhibit longer pauses than tracing collectors.We then analyze reference counting with lazy deletion, the standard technique for avoiding long pauses by deferring deletions and associated reference count decrements, usually to allocation time. ...

Keywords: garbage collection, memory allocation, reference counting, space complexity

7 Type-safe multithreading in cyclone

Dan Grossman
January 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN international workshop on Types in languages design and implementation TLDI '03**, Volume 38 Issue 3
Publisher: ACM Press

Full text available: pdf(228.33 KB) Additional Information: full citation, abstract, references, citings, index terms

We extend Cyclone, a type-safe polymorphic language at the C level of abstraction, with threads and locks. Data races can violate type safety in Cyclone. An extended type system statically guarantees their absence by enforcing that thread-shared data is protected via locking and that thread-local data does not escape the thread that creates it. The extensions interact smoothly with parametric polymorphism and region-based memory management. We present a formal abstract machine that models the ne ...

Keywords: cyclone, data races, types

8 Machine machinery: Quantifying the performance of garbage collection vs. explicit memory management
Matthew Hertz, Emery D. Berger
October 2005 **Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming systems languages and applications OOPSLA '05**
Publisher: ACM Press
Full text available: pdf(1.51 MB) Additional Information: full citation, abstract, references, index terms

Garbage collection yields numerous software engineering benefits, but its quantitative impact on performance remains elusive. One can compare the cost of *conservative* garbage collection to explicit memory management in C/C++ programs by linking in an appropriate collector. This kind of direct comparison is not possible for languages designed for garbage collection (e.g., Java), because programs in these languages naturally do not contain calls to free. Thus, the actual gap between the tim ...

Keywords: explicit memory management, garbage collection, oracular memory management, paging, performance analysis, throughput, time-space tradeoff

9 Session 1: Safe memory reclamation for dynamic lock-free objects using atomic reads and writes
Maged M. Michael
July 2002 **Proceedings of the twenty-first annual symposium on Principles of distributed computing**
Publisher: ACM Press
Full text available: pdf(1.13 MB) Additional Information: full citation, abstract, references, citings

A major obstacle to the wide use of lock-free data structures, despite their many performance and reliability advantages, is the absence of a practical lock-free method for reclaiming the memory of dynamic nodes removed from dynamic lock-free objects for arbitrary reuse.The only prior lock-free memory reclamation method depends on the DCAS atomic primitive, which is not supported on any current processor architecture. Other memory management methods are blocking, require special operating system ...

10 Pointer and escape analysis for multithreaded programs
Alexandru Salcianu, Martin Rinard
June 2001 **ACM SIGPLAN Notices , Proceedings of the eighth ACM SIGPLAN symposium on Principles and practices of parallel programming PPoPP '01**, Volume 36 Issue 7
Publisher: ACM Press
Full text available: pdf(318.11 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new combined pointer and escape analysis for multithreaded programs. The algorithm uses a new abstraction called parallel interaction graphs to analyze the interactions between threads and extract precise points-to, escape, and action ordering information for objects accessed by multiple threads. The analysis is compositional, analyzing each method or thread once to extract a parameterized analysis result that can be specialized for use in any context. ...

11 Stack allocation and synchronization optimizations for Java using escape analysis

Jong-Deok Choi, Manish Gupta, Mauricio J. Serrano, Vugranam C. Sreedhar, Samuel P. Midkiff

November 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 6

Publisher: ACM Press

Full text available: pdf(632.85 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This article presents an *escape analysis* framework for Java to determine (1) if an object is not reachable after its method of creation returns, allowing the object to be allocated on the stack, and (2) if an object is reachable only from a single thread during its lifetime, allowing unnecessary synchronization operations on that object to be removed. We introduce a new program abstraction for escape analysis, the *connection graph*, that is used to establish reachability relationshi ...

Keywords: Connection graphs, escape analysis, points-to graph

12 Systems and prototypes: Java support for data-intensive systems: experiences building the telegraph dataflow system

Mehul A. Shah, Michael J. Franklin, Samuel Madden, Joseph M. Hellerstein

December 2001 **ACM SIGMOD Record**, Volume 30 Issue 4

Publisher: ACM Press

Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings

Database system designers have traditionally had trouble with the default services and interfaces provided by operating systems. In recent years, developers and enthusiasts have increasingly promoted Java as a serious platform for building data-intensive servers. Java provides a number of very helpful language features, as well as a full run-time environment reminiscent of a traditional operating system. This combination of features and community support raises the question of whether Java is be ...

13 Typed memory management via static capabilities

David Walker, Karl Crary, Greg Morrisett

July 2000 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 22 Issue 4

Publisher: ACM Press

Full text available: pdf(662.98 KB) Additional Information: full citation, abstract, references, citings, index terms

Region-based memory management is an alternative to standard tracing garbage collection that makes operation such as memory deallocation explicit but verifiably safe. In this article, we present a new compiler intermediate language, called the Capability Language (CL), that supports region-based memory management and enjoys a provably safe type systems. Unlike previous region-based type system, region lifetimes need not be lexically scoped, and yet the language may be checked for safety wit ...

Keywords: certified code, region-based memory management, type-directed compilation, typed intermediate languages

14 Mostly lock-free malloc

Dave Dice, Alex Garthwaite
June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management ISMM '02**, Volume 38 Issue 2 supplement
Publisher: ACM Press

Full text available: pdf(609.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Modern multithreaded applications, such as application servers and database engines, can severely stress the performance of user-level memory allocators like the ubiquitous malloc subsystem. Such allocators can prove to be a major scalability impediment for the applications that use them, particularly for applications with large numbers of threads running on high-order multiprocessor systems.This paper introduces Multi-Processor Restartable Critical Sections, or MP-RCS. MP-RCS permits user-level ...

Keywords: affinity, locality, lock-free operations, malloc, restartable critical sections

15 Implementation techniques: Dynamic object sampling for pretenuring
Maria Jump, Stephen M. Blackburn, Kathryn S. McKinley
October 2004 **Proceedings of the 4th international symposium on Memory management**
Publisher: ACM Press
Full text available: pdf(188.82 KB) Additional Information: full citation, abstract, references, index terms

Many state-of-the-art garbage collectors are generational, collecting the young nursery objects more frequently than old objects. These collectors perform well because young objects tend to die at a higher rate than old ones. However, these collectors do not examine object lifetimes with respect to any particular program or allocation site. This paper introduces low-cost object sampling to dynamically determine lifetimes. The sampler marks an object and records its allocation site every n byt ...

Keywords: dynamic pretenuring, garbage collection, memory management, object sampling

16 Memory allocation for long-running server applications
Per-Åke Larson, Murali Krishnan
October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3
Publisher: ACM Press

Full text available: pdf(1.13 MB) Additional Information: full citation, abstract, references, citings, index terms

Prior work on dynamic memory allocation has largely neglected long-running server applications, for example, web servers and mail servers. Their requirements differ from those of one-shot applications like compilers or text editors. We investigated how to build an allocator that is not only fast and memory efficient but also scales well on SMP machines. We found that it is not sufficient to focus on reducing lock contention - higher speedups require a reduction in cache misses and bus traffic. W ...

Keywords: cache-conscious algorithms, concurrency, dynamic memory allocation, multiprocessor scalability, reducing lock contention, server applications

17 Concurrency: Write barrier elision for concurrent garbage collectors
Martin T. Vechev, David F. Bacon
October 2004 **Proceedings of the 4th international symposium on Memory management**
Publisher: ACM Press
Full text available: pdf(490.73 KB) Additional Information: full citation, abstract, references, index terms

Concurrent garbage collectors require write barriers to preserve consistency, but these barriers impose significant direct and indirect costs. While there has been a lot of work on optimizing write barriers, we present the first study of their elision in a concurrent collector. We show conditions under which write barriers are redundant, and describe how these conditions can be applied to both incremental update or snapshot-at-the-beginning barriers. We then evaluate the potential for write b ...

Keywords: concurrent garbage collection, write barrier

18 Automated discovery of scoped memory regions for real-time Java

Morgan Deters, Ron K. Cytron

June 2002 **ACM SIGPLAN Notices , Proceedings of the 3rd international symposium on Memory management ISMM '02**, Volume 38 Issue 2 supplement

Publisher: ACM Press

Full text available: pdf(227.49 KB) Additional Information: full citation, abstract, references, citings, index terms

Advances in operating systems and languages have brought the ideal of reasonably-bounded execution time closer to developers who need such assurances for real-time and embedded systems applications. Recently, extensions to the Java libraries and virtual machine have been proposed in an emerging standard, which provides for specification of release times, execution costs, and deadlines for a restricted class of threads. To use such features, the code executing in the thread must never reference s ...

Keywords: garbage collection, memory management, real-time Java, regions, trace-based analysis

19 A dynamic memory management unit for embedded real-time system-on-a-chip

Mohamed Shalan, Vincent J. Mooney

November 2000 **Proceedings of the 2000 international conference on Compilers, architecture, and synthesis for embedded systems**

Publisher: ACM Press

Full text available: pdf(321.80 KB) Additional Information: full citation, citings

Keywords: SoCDMMU, dynamic memory management, embedded systems, real-time systems, system-on-a-chip, two-level memory management

20 The duality of memory and communication in the implementation of a multiprocessor operating system

M. Young, A. Tevanian, R. Rashid, D. Golub, J. Eppinger

November 1987 **ACM SIGOPS Operating Systems Review , Proceedings of the eleventh ACM Symposium on Operating systems principles SOSP '87**, Volume 21 Issue 5

Publisher: ACM Press

Full text available: pdf(1.26 MB) Additional Information: full citation, abstract, references, citings, index terms

Mach is a multiprocessor operating system being implemented at Carnegie-Mellon University. An important component of the Mach design is the use of memory objects which can be managed either by the kernel or by user programs through a message interface. This feature allows applications such as transaction management systems to participate in decisions regarding secondary storage management and page replacement.This paper explores the goals, design and implementation of Mach and it ...

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Thu, 3 Aug 2006, 9:47:36 PM EST

Search Query Display

[redacted]

Recent Search Queries

#1 ((~~process communication~~ <and> ~~memory management~~)<in>metadata)

#2 ((~~process communication~~ <and> ~~memory management~~)<in>metadata)

#3 ((process communication <and> memory allocation)<in>metadata)

[redacted]